APRENDA PYTHON
por Tito Petri
arte por DadoAlmeida

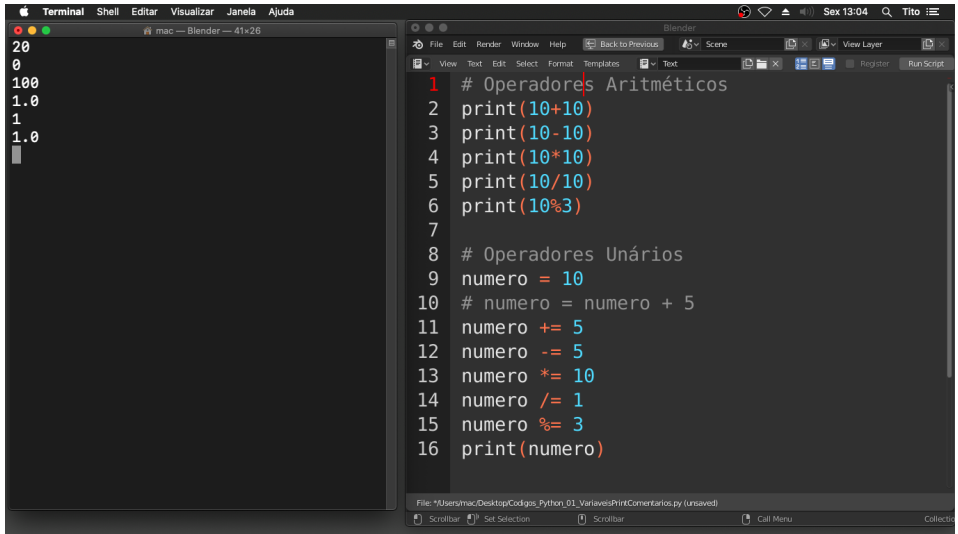
Apostila de Python para Blender

por Tito Petri

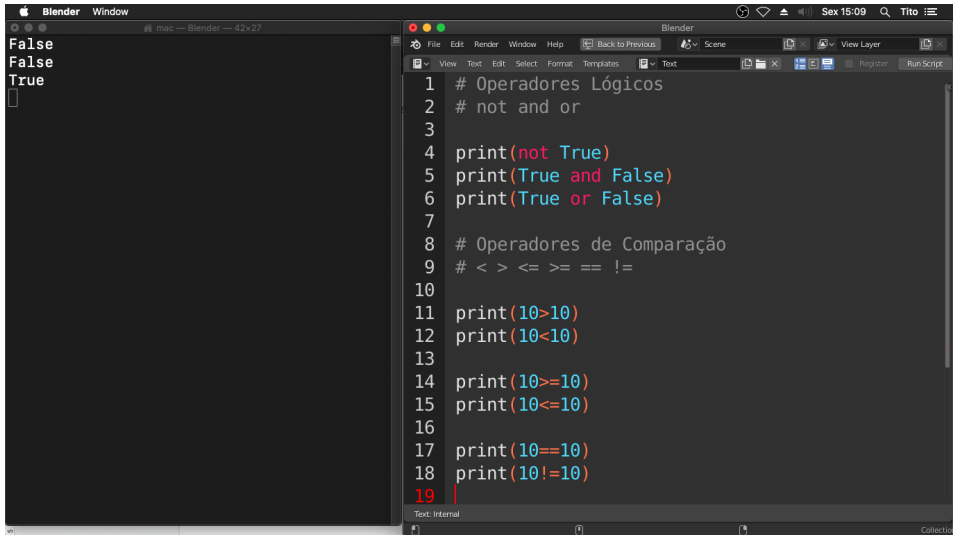
Primeira edição, 2022
ISBN 0-5487263-1-5
www.titopetri.com

Ferramentas e Editores para Programar

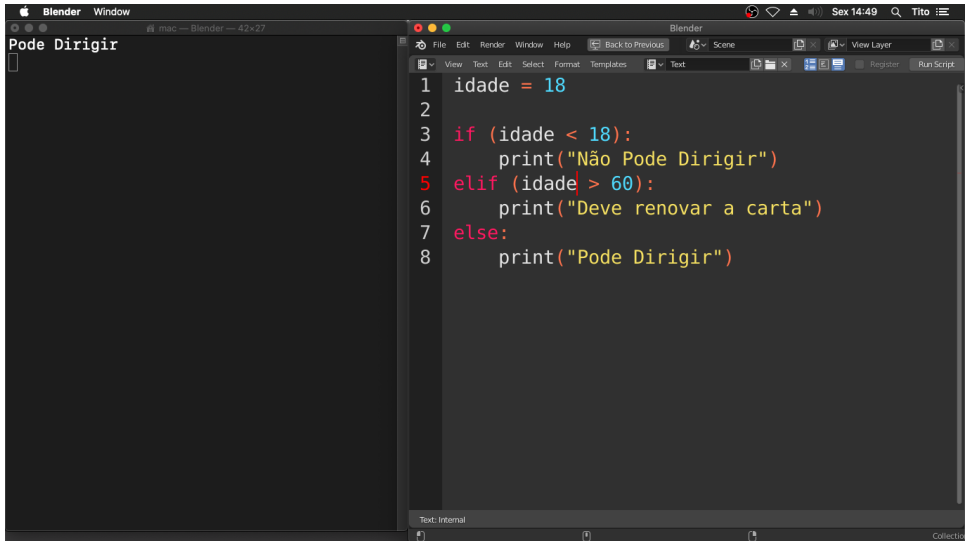
Print e Declaração de Variáveis

```
print("Bem-Vindo ao Blender Python!")

# isto é um comentário

"""isto é um comentário
feito em
múltiplas linhas!"""

nome = "Felpudo Jr."
print(nome)

numero = 10
altura = 1.75
estado = False
lista = [10,50,100,1000]
nomes = ["Tito", "Felpudo", "Fofura"]

print(nomes)
```

Operadores Aritméticos e Unários

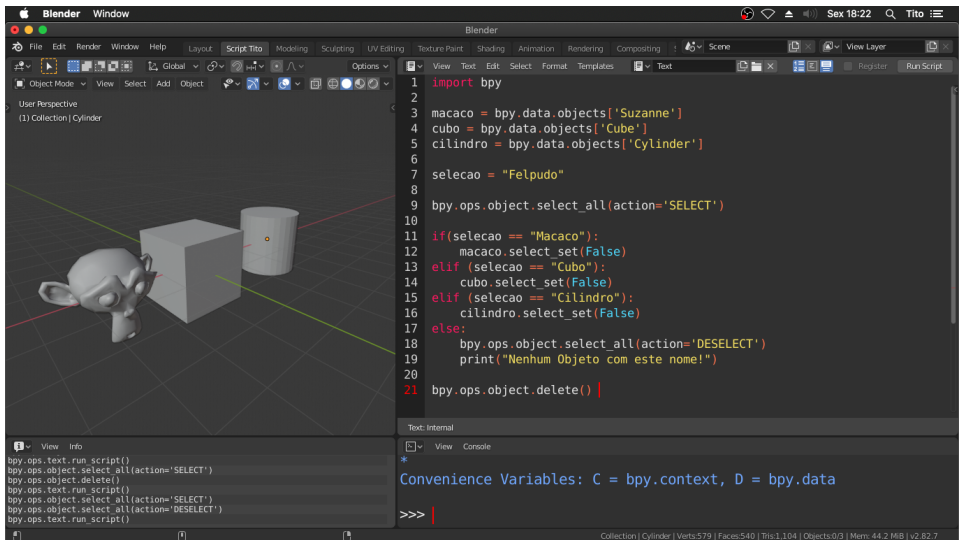
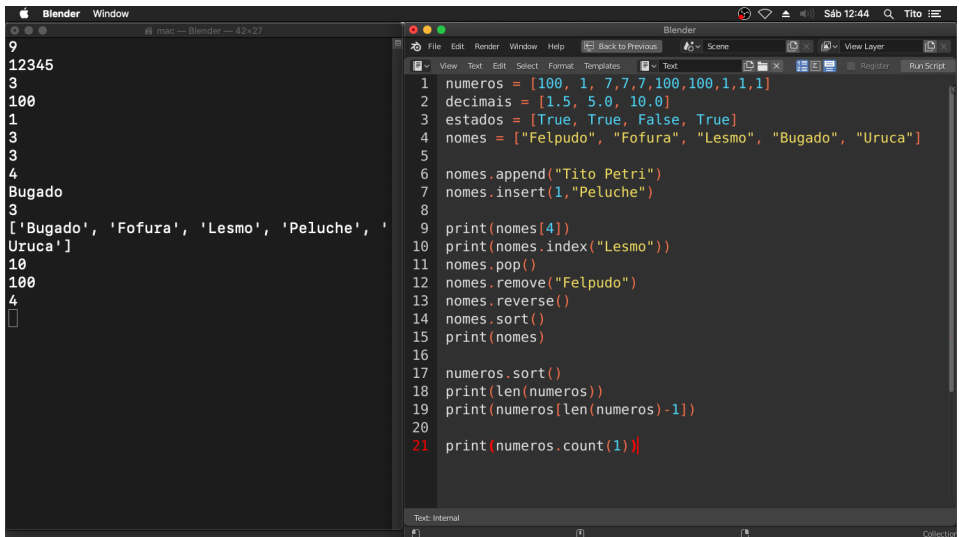
```
# Operadores Aritméticos
print(10+10)
print(10-10)
print(10*10)
print(10/10)
print(10%3)

# Operadores Unários
numero = 10
# numero = numero + 5
numero += 5
numero -= 5
numero *= 10
numero /= 1
numero %= 3
print(numero)
```

Funções Matemáticas

```
import math as m

print(m.pi*2)

print(m.sin(90))
print(m.cos(90))
print(m.tan(90))

print(m.floor(100/3))
print(m.ceil(100/3))
```

Operadores Lógicos e de Comparação

```
# Operadores Lógicos
# not and or

print(not True)
print(True and False)
print(True or False)

# Operadores de Comparação
# < > <= >= == !=

print(10>10)
print(10<10)

print(10>=10)
print(10<=10)

print(10==10)
print(10!=10)
```

Condição If Elif e Else

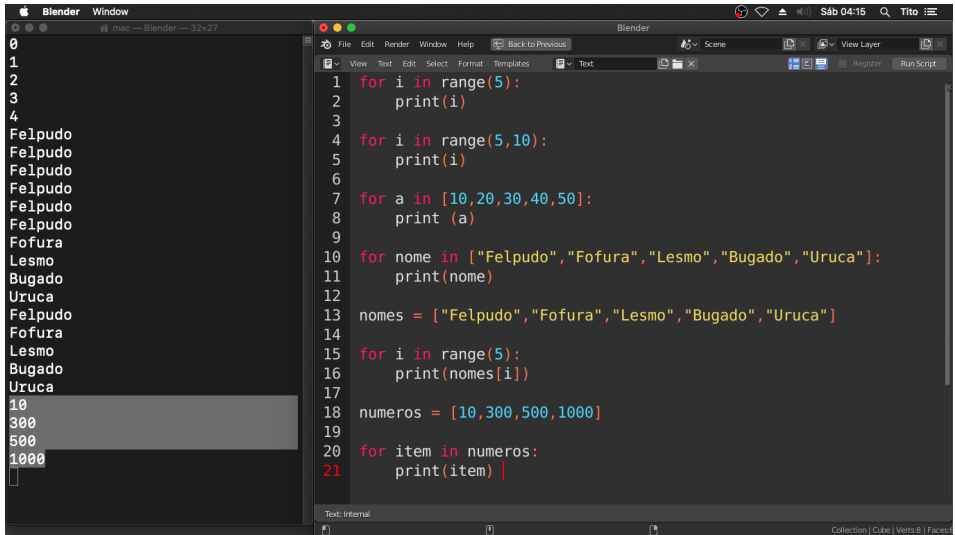
```
idade = 18

if (idade < 18):
    print("Não Pode Dirigir")
elif (idade > 60):
    print("Deve renovar a carta")
else:
    print("Pode Dirigir")
```

Operações com Objetos da Cena

```
import bpy

macaco = bpy.data.objects['Suzanne']
cubo = bpy.data.objects['Cube']
cilindro = bpy.data.objects['Cylinder']

selecao = "Felpudo"

bpy.ops.object.select_all(action='SELECT')

if(selecao == "Macaco"):
    macaco.select_set(False)
elif (selecao == "Cubo"):
    cubo.select_set(False)
elif (selecao == "Cilindro"):
    cilindro.select_set(False)
else:
    bpy.ops.object.select_all(action='DESELECT')
```

```
    print("Nenhum Objeto com este nome!")

bpy.ops.object.delete()
```

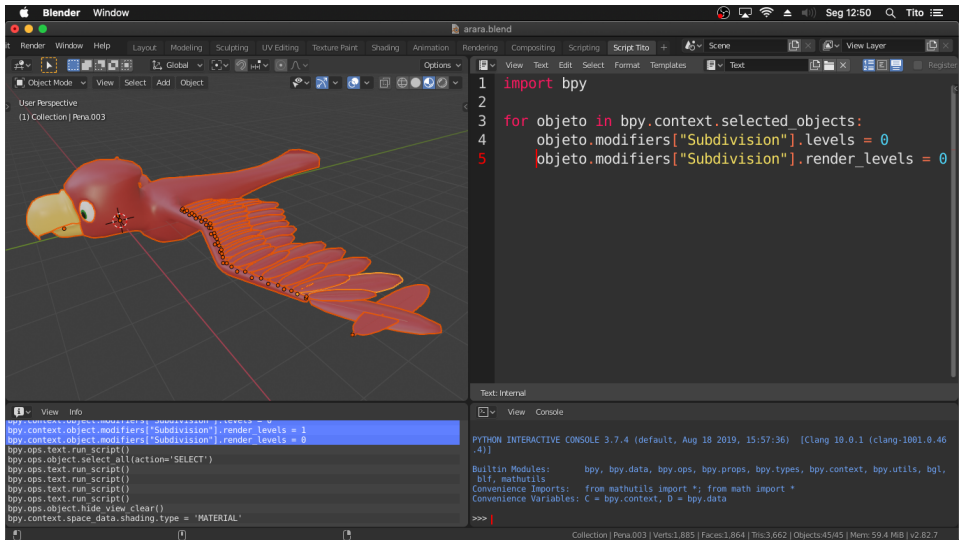
Listas ou Arrays

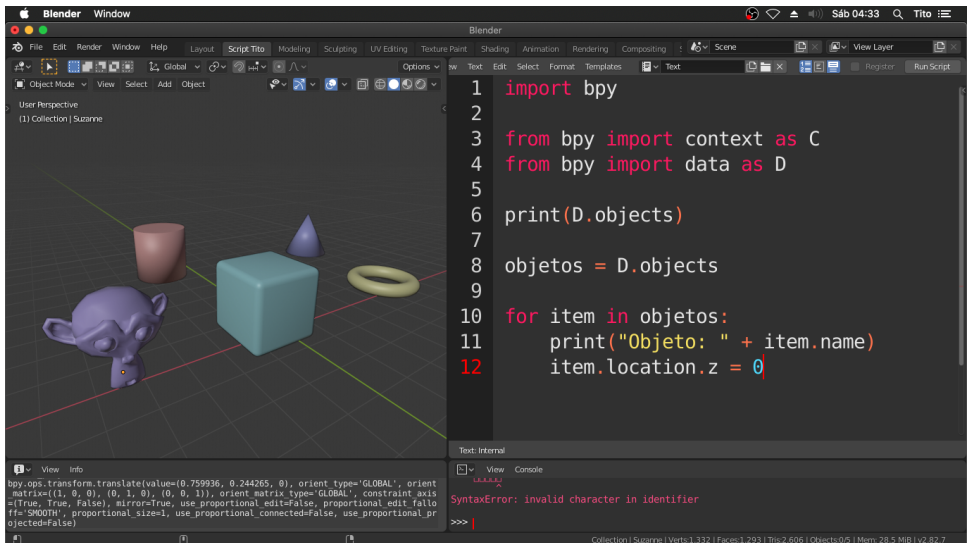
```
numeros = [100, 1, 7,7,7,100,100,1,1,1]
decimais = [1.5, 5.0, 10.0]
estados = [True, True, False, True]
nomes = ["Felpudo", "Fofura", "Lesmo", "Bugado", "Uruca"]

nomes.append("Tito Petri")
nomes.insert(1,"Peluche")

print(nomes[4])
print(nomes.index("Lesmo"))
nomes.pop()
nomes.remove("Felpudo")
nomes.reverse()
nomes.sort()
print(nomes)

numeros.sort()
print(len(numeros))
```

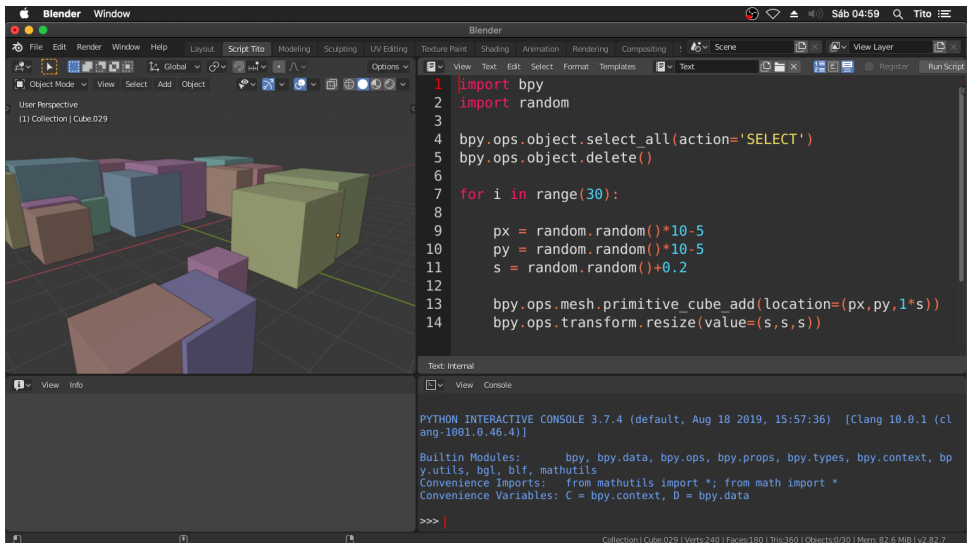
```
print(numeros[len(numeros)-1])

print(numeros.count(1))
```

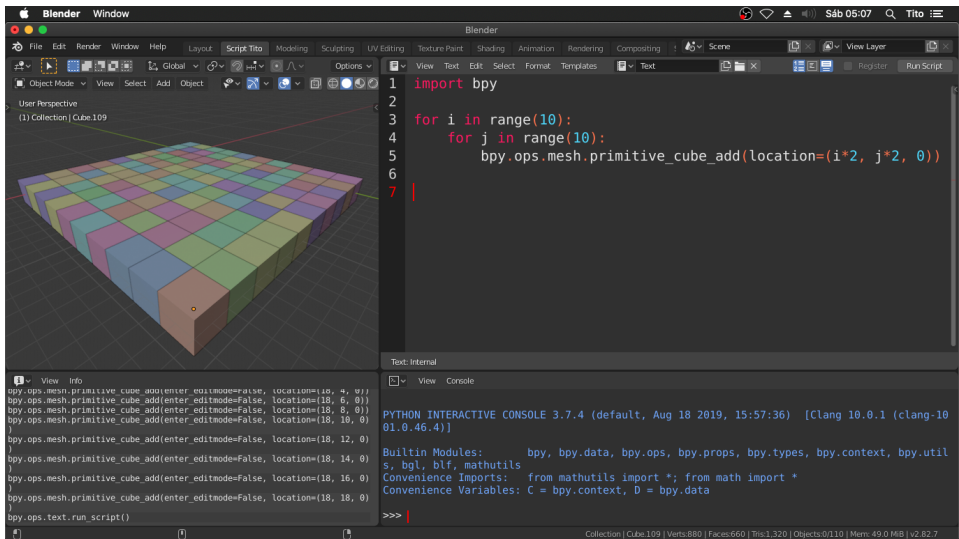
Loop de Repetição For

```
for i in range(5):
    print(i)

for i in range(5,10):
    print(i)

for a in [10,20,30,40,50]:
    print (a)

for nome in ["Felpudo","Fofura","Lesmo","Bugado","Uruca"]:
    print(nome)

nomes = ["Felpudo","Fofura","Lesmo","Bugado","Uruca"]

for i in range(5):
    print(nomes[i])

numeros = [10,300,500,1000]
```

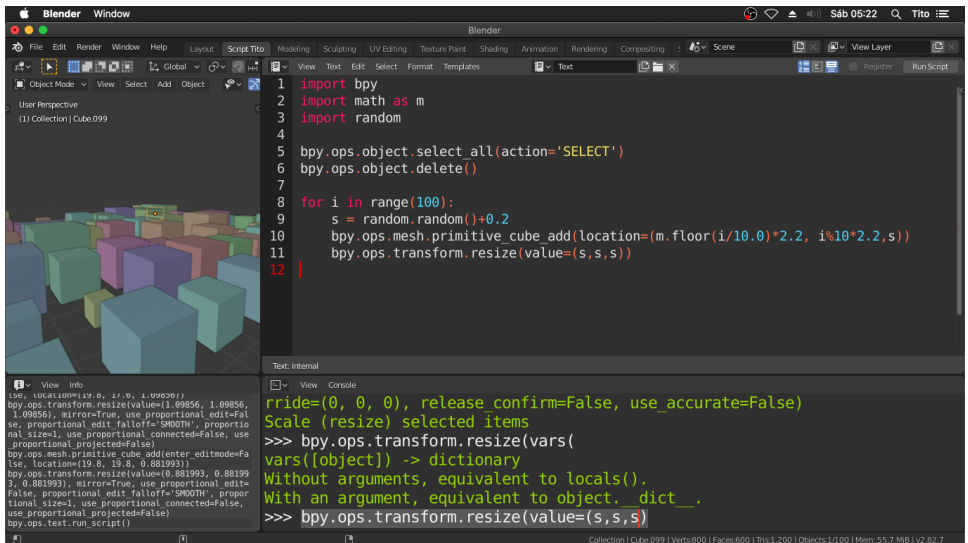
```
for item in numeros:
    print(item)
```

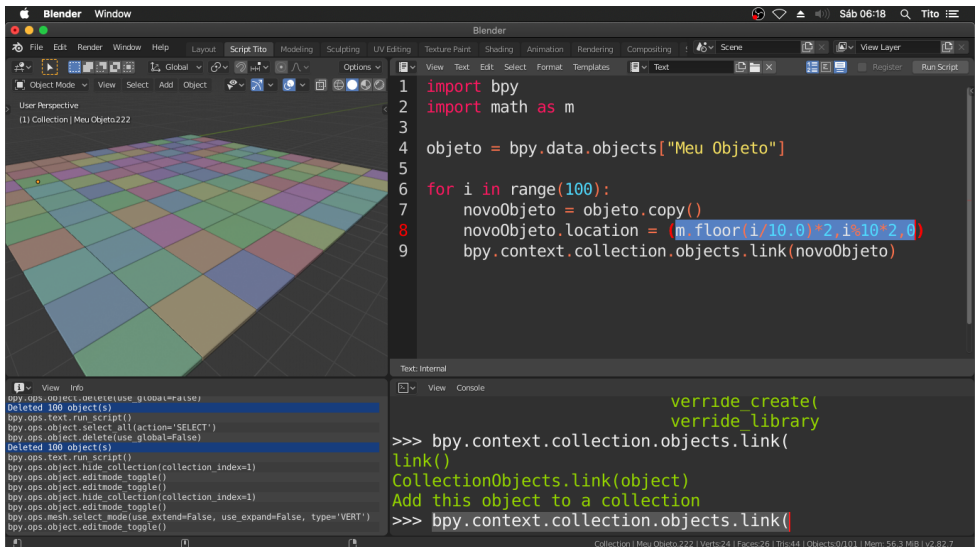
Percorrendo Objetos Selecionados

```
import bpy

for objeto in bpy.context.selected_objects:
    objeto.modifiers["Subdivision"].levels = 0
    objeto.modifiers["Subdivision"].render_levels = 0
```

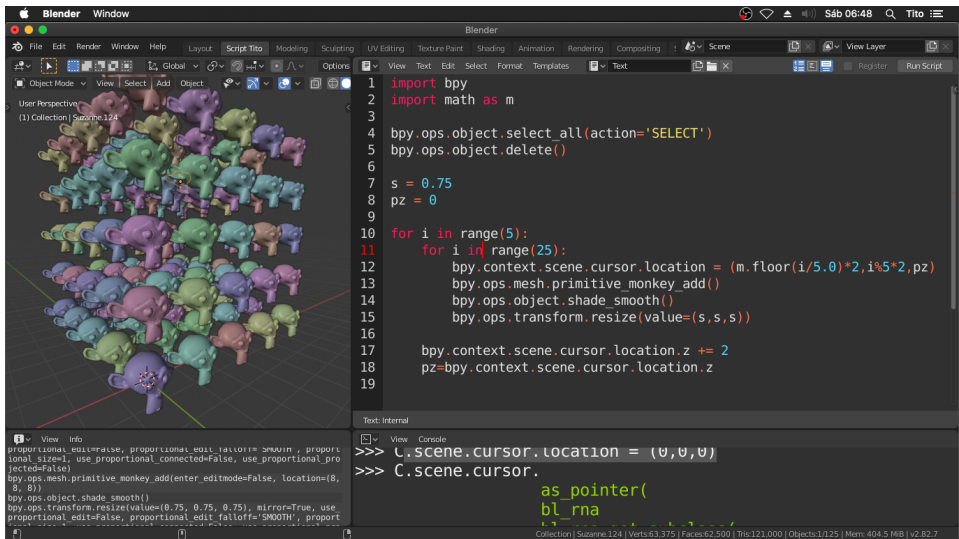
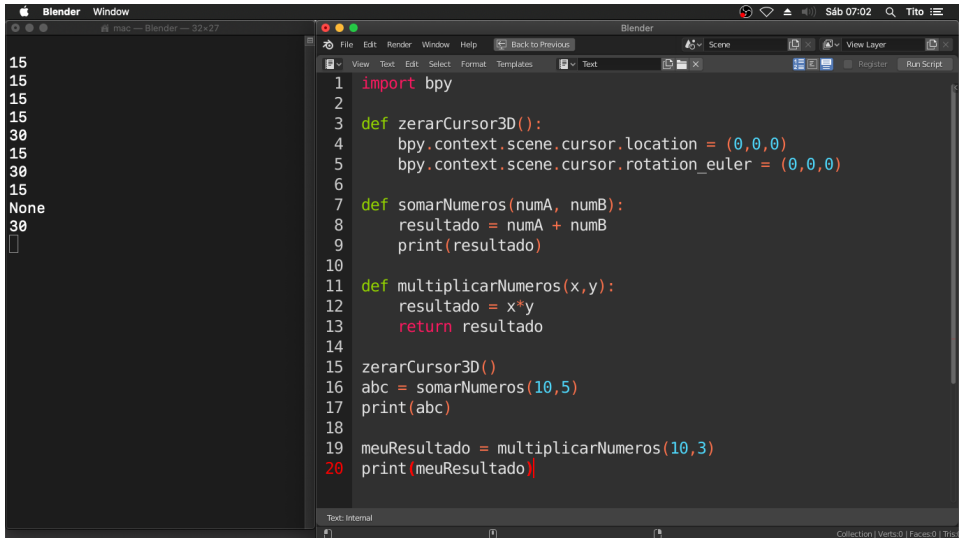
Percorrendo Objetos na Cena

```
import bpy

from bpy import context as C
from bpy import data as D

print(D.objects)

objetos = D.objects

for item in objetos:
    print("Objeto: " + item.name)
    item.location.z = 0
```

Números Aleatórios - Random

```
import bpy
import random

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

for i in range(30):

    px = random.random()*10-5
    py = random.random()*10-5
    s = random.random()+0.2

    bpy.ops.mesh.primitive_cube_add(location=(px,py,1*s))
    bpy.ops.transform.resize(value=(s,s,s))
```

Loops Aninhados e Matriz 2D

```
import bpy

for i in range(10):
    for j in range(10):
        bpy.ops.mesh.primitive_cube_add(location=(i*2, j*2, 0))
```

Matriz 2D de Objetos com Math

```
import bpy
import math as m
import random

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

for i in range(100):
    s = random.random()+0.2
    bpy.ops.mesh.primitive_cube_add(location=(m.floor(i/10.0)*2.2, i%10*2.2,s))
    bpy.ops.transform.resize(value=(s,s,s))
```

Replicando Objeto em uma Matriz 2D

```
import bpy
import math as m

objeto = bpy.data.objects["Meu Objeto"]

for i in range(100):
    novoObjeto = objeto.copy()
    novoObjeto.location = (m.floor(i/10.0)*2,i%10*2,0)
    bpy.context.collection.objects.link(novoObjeto)
```

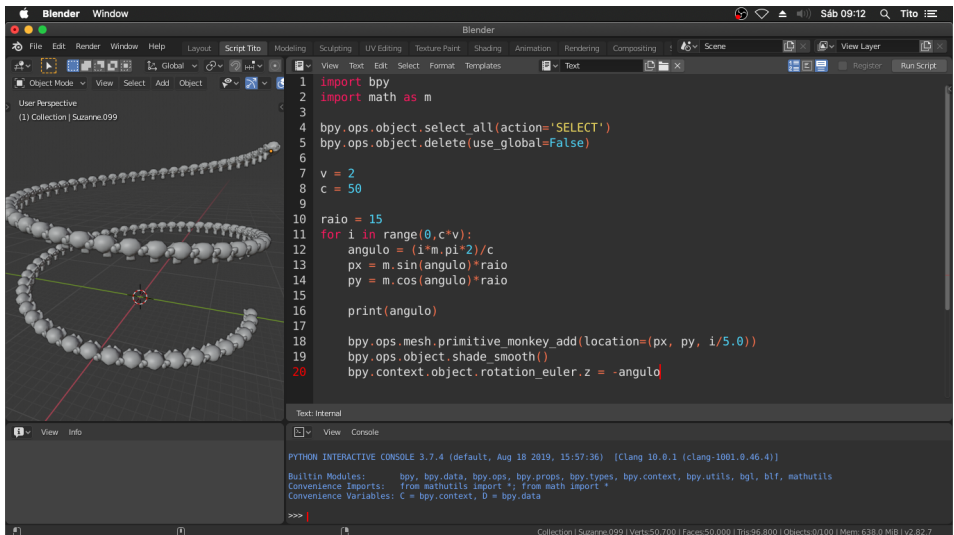
Matriz de Objetos em 3 Dimensões

```
import bpy
import math as m

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

s = 0.75
pz = 0

for i in range(5):
    for i in range(25):
        bpy.context.scene.cursor.location = (m.floor(i/5.0)*2,i%5*2,pz)
        bpy.ops.mesh.primitive_monkey_add()
        bpy.ops.object.shade_smooth()
        bpy.ops.transform.resize(value=(s,s,s))

    bpy.context.scene.cursor.location.z += 2
```

```
pz=bpy.context.scene.cursor.location.z
```

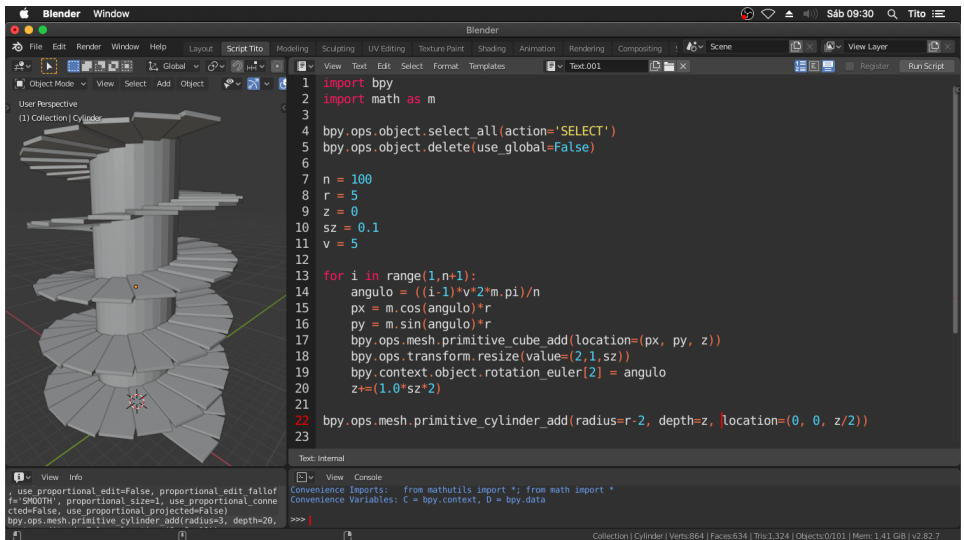
Métodos Funções e Procedimentos

```
import bpy

def zerarCursor3D():
    bpy.context.scene.cursor.location = (0,0,0)
    bpy.context.scene.cursor.rotation_euler = (0,0,0)

def somarNumeros(numA, numB):
    resultado = numA + numB
    print(resultado)

def multiplicarNumeros(x,y):
    resultado = x*y
    return resultado

zerarCursor3D()
abc = somarNumeros(10,5)
print(abc)
```

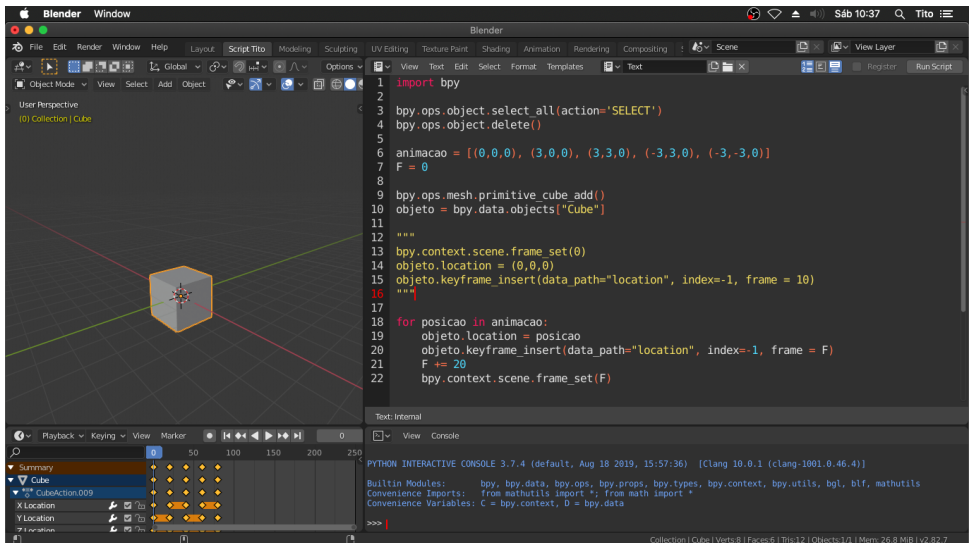
```
meuResultado = multiplicarNumeros(10,3)
print(meuResultado)
```

Replicando Objetos em Formato Circular com Seno e Cosseno

```
import bpy
import math as m

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete(use_global=False)

v = 2
c = 50

raio = 15
for i in range(0,c*v):
    angulo = (i*m.pi*2)/c
    px = m.sin(angulo)*raio
    py = m.cos(angulo)*raio
```

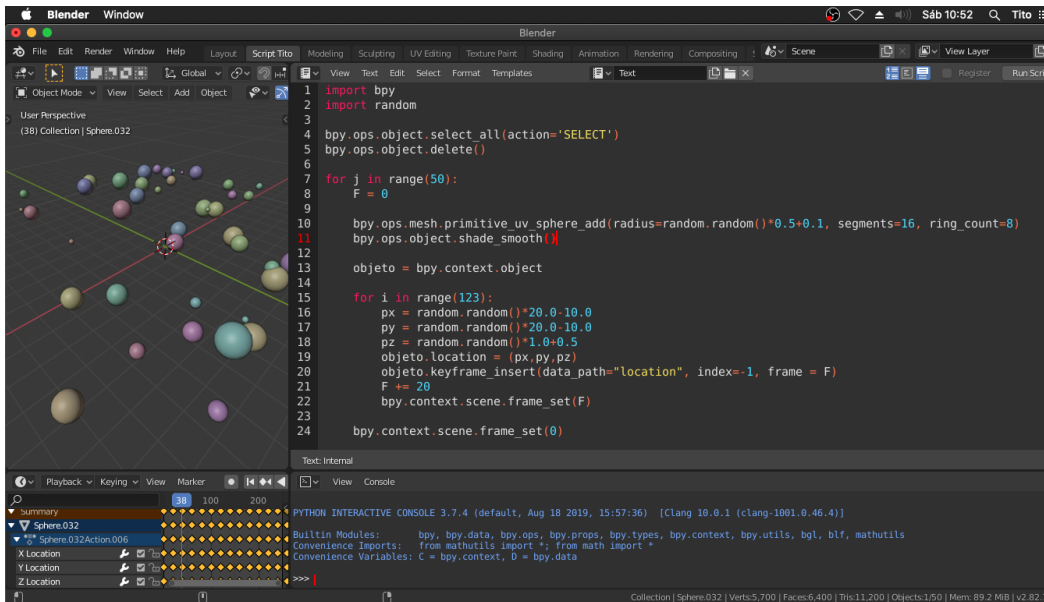
```
print(angulo)

bpy.ops.mesh.primitive_monkey_add(location=(px, py, i/5.0))
bpy.ops.object.shade_smooth()
bpy.context.object.rotation_euler.z = -angulo
```

Criando uma Escada em Espiral

```
import bpy
import math as m

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete(use_global=False)

n = 100
r = 5
z = 0
sz = 0.1
v = 5

for i in range(1,n+1):
    angulo = ((i-1)*v*2*m.pi)/n
    px = m.cos(angulo)*r
    py = m.sin(angulo)*r
    bpy.ops.mesh.primitive_cube_add(location=(px, py, z))
    bpy.ops.transform.resize(value=(2,1,sz))
```

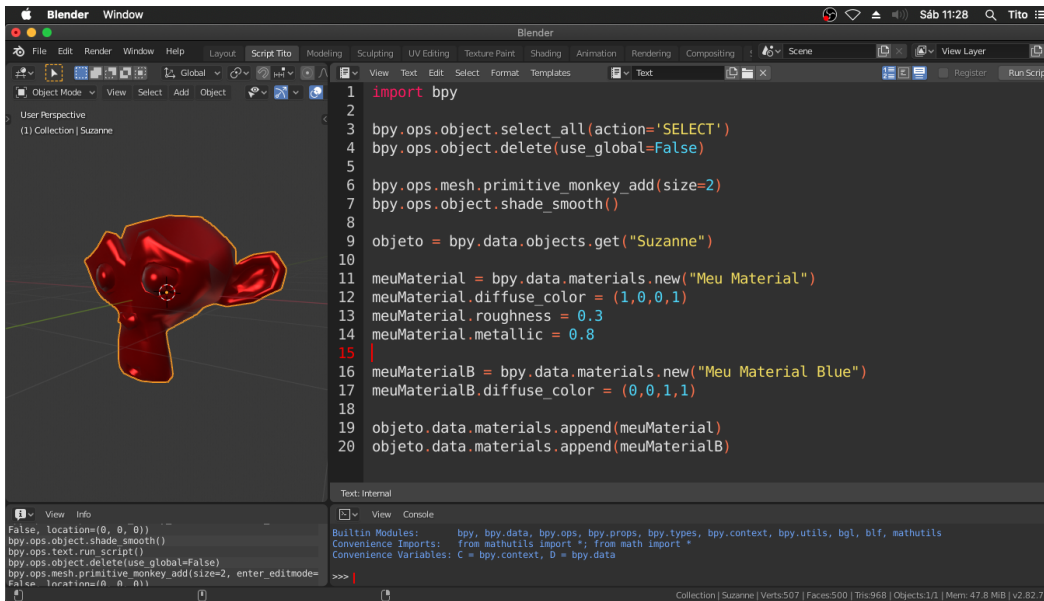
```
    bpy.context.object.rotation_euler[2] = angulo
    z+=(1.0*sz*2)

bpy.ops.mesh.primitive_cylinder_add(radius=r-2, depth=z, location=(0, 0, z/2))
```

Inserindo Keyframes de Animação

```
import bpy

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

animacao = [(0,0,0), (3,0,0), (3,3,0), (-3,3,0), (-3,-3,0)]
F = 0

bpy.ops.mesh.primitive_cube_add()
objeto = bpy.data.objects["Cube"]

"""
bpy.context.scene.frame_set(0)
objeto.location = (0,0,0)
objeto.keyframe_insert(data_path="location", index=-1, frame = 10)
"""

for posicao in animacao:
```

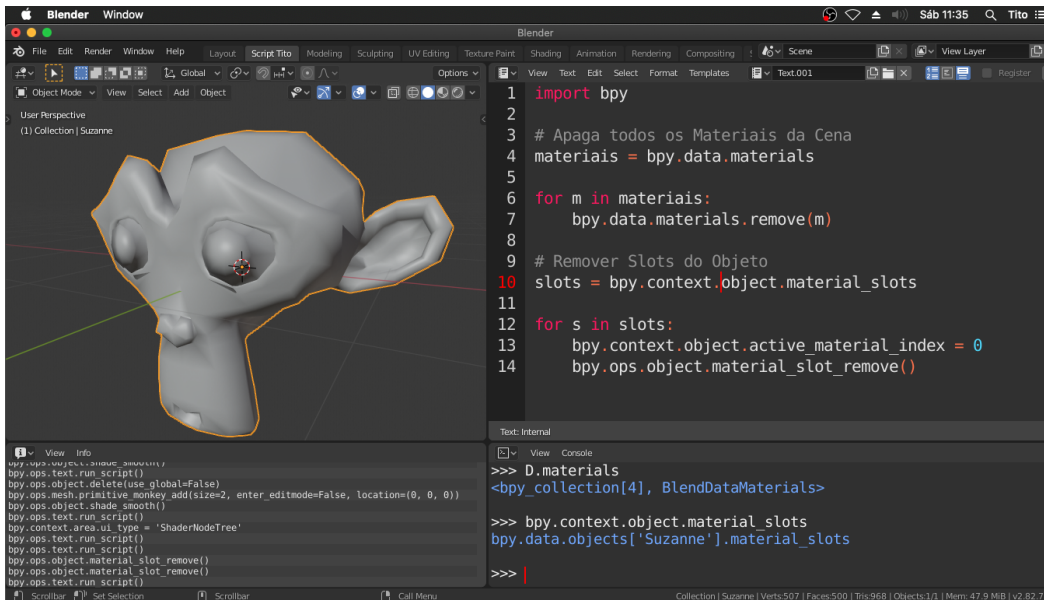
```
objeto.location = posicao
objeto.keyframe_insert(data_path="location", index=-1, frame = F)
F += 20
bpy.context.scene.frame_set(F)
```

Animando um Objeto Aleatoriamente

```
import bpy
import random

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

for j in range(50):
    F = 0

    bpy.ops.mesh.primitive_uv_sphere_add(radius=random.random()*0.5+0.1, segments=16,
ring_count=8)
    bpy.ops.object.shade_smooth()

    objeto = bpy.context.object

    for i in range(123):
```

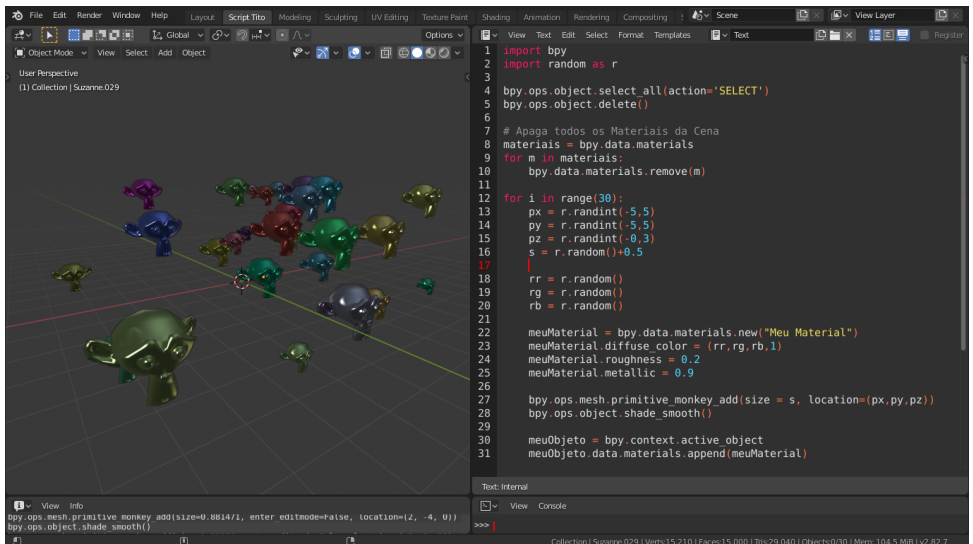
```
    px = random.random()*20.0-10.0
    py = random.random()*20.0-10.0
    pz = random.random()*1.0+0.5
    objeto.location = (px,py,pz)
    objeto.keyframe_insert(data_path="location", index=-1, frame = F)
    F += 20
    bpy.context.scene.frame_set(F)

bpy.context.scene.frame_set(0)
```

Criando Materiais Programaticamente

```
import bpy

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete(use_global=False)

bpy.ops.mesh.primitive_monkey_add(size=2)
bpy.ops.object.shade_smooth()

objeto = bpy.data.objects.get("Suzanne")

meuMaterial = bpy.data.materials.new("Meu Material")
meuMaterial.diffuse_color = (1,0,0,1)
meuMaterial.roughness = 0.3
meuMaterial.metallic = 0.8

meuMaterialB = bpy.data.materials.new("Meu Material Blue")
```

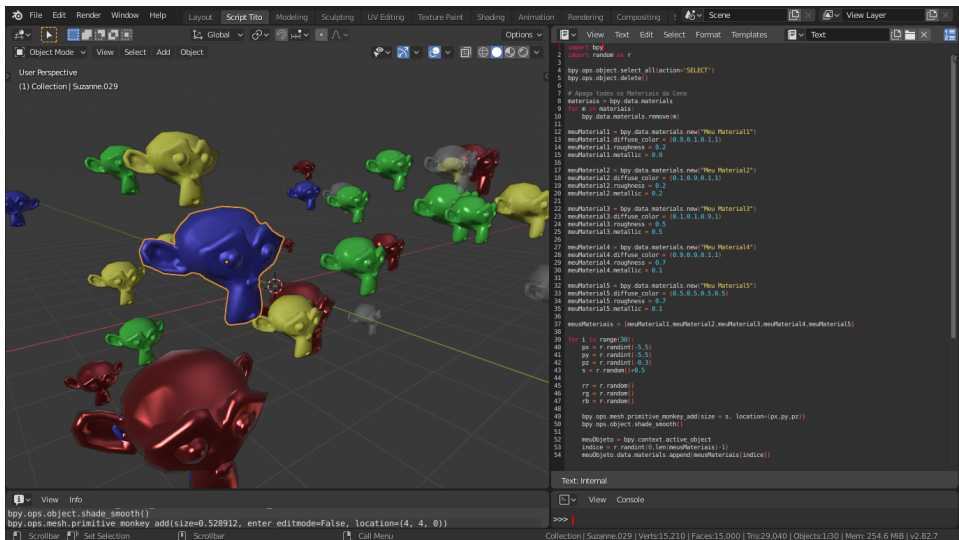
```
meuMaterialB.diffuse_color = (0,0,1,1)

objeto.data.materials.append(meuMaterial)
objeto.data.materials.append(meuMaterialB)
```

Método para Apagar Biblioteca e Slots de Materiais

```
import bpy

# Apaga todos os Materiais da Cena
materiais = bpy.data.materials

for m in materiais:
    bpy.data.materials.remove(m)

# Remover Slots do Objeto
slots = bpy.context.object.material_slots

for s in slots:
    bpy.context.object.active_material_index = 0
```

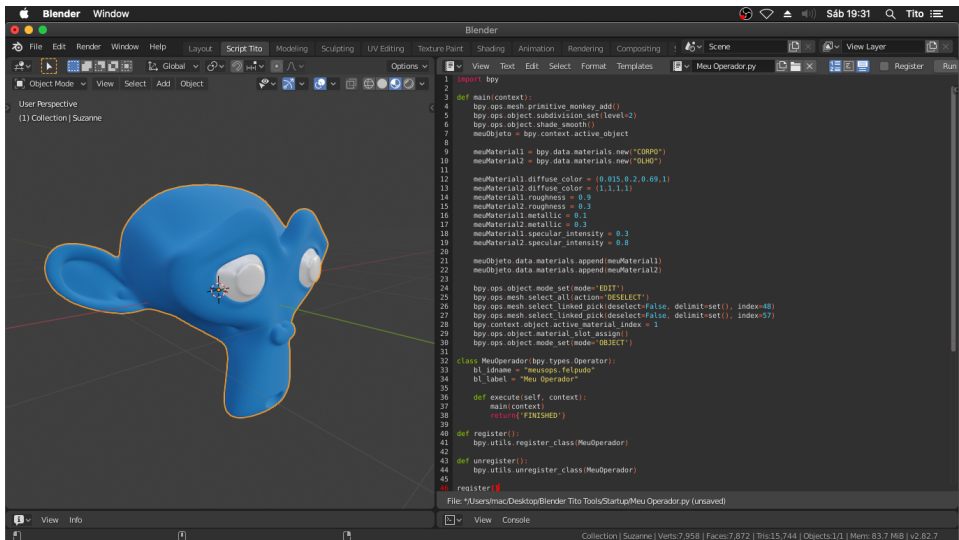
```
bpy.ops.object.material_slot_remove()
```

Criando Múltiplos Materiais Aleatoriamente

```
import bpy
import random as r

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

# Apaga todos os Materiais da Cena
materiais = bpy.data.materials
for m in materiais:
    bpy.data.materials.remove(m)

for i in range(30):
    px = r.randint(-5,5)
    py = r.randint(-5,5)
    pz = r.randint(-0,3)
```

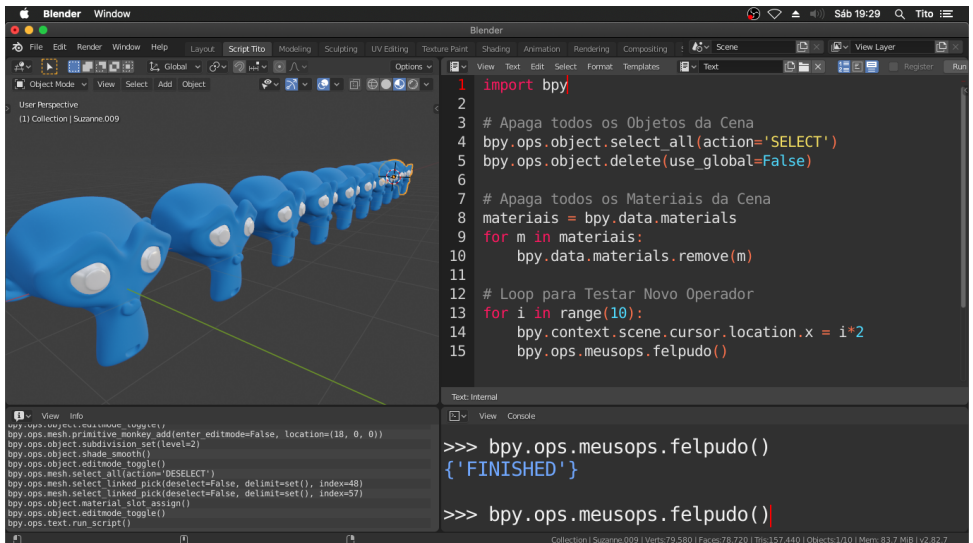
```
s = r.random()+0.5

rr = r.random()
rg = r.random()
rb = r.random()

meuMaterial = bpy.data.materials.new("Meu Material")
meuMaterial.diffuse_color = (rr,rg,rb,1)
meuMaterial.roughness = 0.2
meuMaterial.metallic = 0.9

bpy.ops.mesh.primitive_monkey_add(size = s, location=(px,py,pz))
bpy.ops.object.shade_smooth()

meuObjeto = bpy.context.active_object
meuObjeto.data.materials.append(meuMaterial)
```

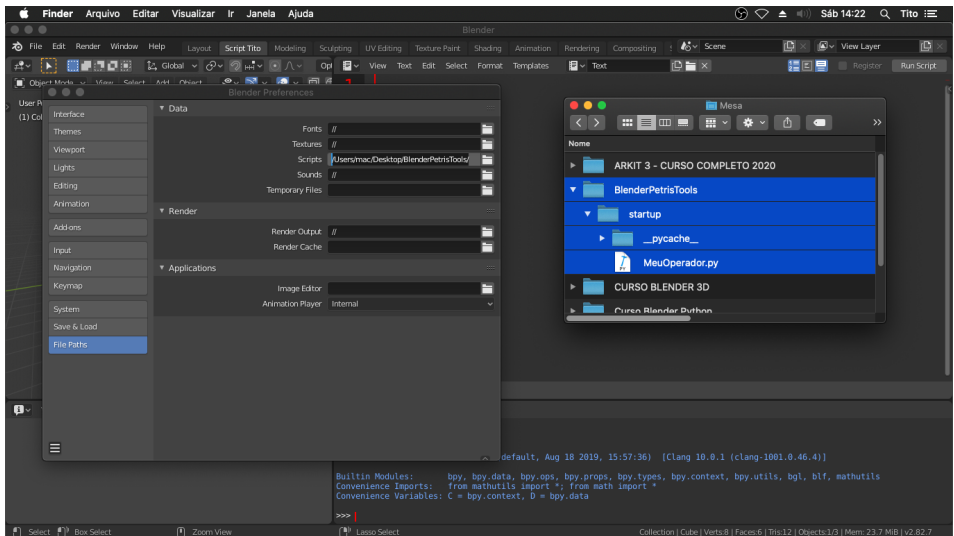
Distribuindo Materiais Aleatoriamente

```
import bpy
import random as r

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

# Apaga todos os Materiais da Cena
materiais = bpy.data.materials
for m in materiais:
    bpy.data.materials.remove(m)

meuMaterial1 = bpy.data.materials.new("Meu Material1")
meuMaterial1.diffuse_color = (0.9,0.1,0.1,1)
meuMaterial1.roughness = 0.2
meuMaterial1.metallic = 0.9

meuMaterial2 = bpy.data.materials.new("Meu Material2")
meuMaterial2.diffuse_color = (0.1,0.9,0.1,1)
```

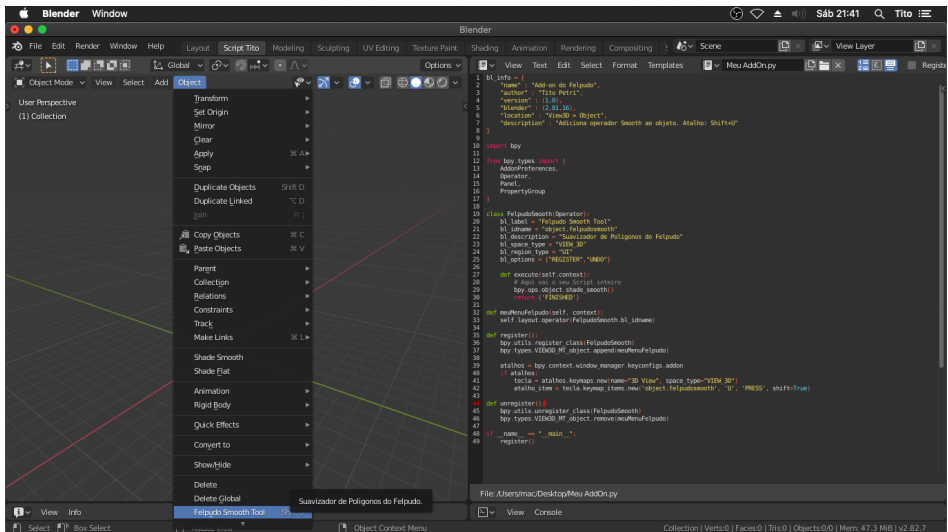
```
meuMaterial2.roughness = 0.2
meuMaterial2.metallic = 0.2

meuMaterial3 = bpy.data.materials.new("Meu Material3")
meuMaterial3.diffuse_color = (0.1,0.1,0.9,1)
meuMaterial3.roughness = 0.5
meuMaterial3.metallic = 0.5

meuMaterial4 = bpy.data.materials.new("Meu Material4")
meuMaterial4.diffuse_color = (0.9,0.9,0.1,1)
meuMaterial4.roughness = 0.7
meuMaterial4.metallic = 0.1

meuMaterial5 = bpy.data.materials.new("Meu Material5")
meuMaterial5.diffuse_color = (0.5,0.5,0.5,0.5)
meuMaterial5.roughness = 0.7
meuMaterial5.metallic = 0.1

meusMateriais = [meuMaterial1,meuMaterial2,meuMaterial3,meuMaterial4,meuMaterial5]

for i in range(30):
    px = r.randint(-5,5)
    py = r.randint(-5,5)
    pz = r.randint(-0,3)
    s = r.random()+0.5

    rr = r.random()
    rg = r.random()
    rb = r.random()

    bpy.ops.mesh.primitive_monkey_add(size = s, location=(px,py,pz))
    bpy.ops.object.shade_smooth()

    meuObjeto = bpy.context.active_object
    indice = r.randint(0,len(meusMateriais)-1)
    meuObjeto.data.materials.append(meusMateriais[indice])
```

Criando e Registrando um Operador - Objeto com Múltiplos Materiais

```
import bpy

def main(context):
    bpy.ops.mesh.primitive_monkey_add()
    bpy.ops.object.subdivision_set(level=2)
    bpy.ops.object.shade_smooth()
    meuObjeto = bpy.context.active_object

    meuMaterial1 = bpy.data.materials.new("CORPO")
    meuMaterial2 = bpy.data.materials.new("OLHO")

    meuMaterial1.diffuse_color = (0.015,0.2,0.69,1)
    meuMaterial2.diffuse_color = (1,1,1,1)
    meuMaterial1.roughness = 0.9
    meuMaterial2.roughness = 0.3
```

```
    meuMaterial1.metallic = 0.1
    meuMaterial2.metallic = 0.3
    meuMaterial1.specular_intensity = 0.3
    meuMaterial2.specular_intensity = 0.8

    meuObjeto.data.materials.append(meuMaterial1)
    meuObjeto.data.materials.append(meuMaterial2)

    bpy.ops.object.mode_set(mode='EDIT')
    bpy.ops.mesh.select_all(action='DESELECT')
    bpy.ops.mesh.select_linked_pick(deselect=False, delimit=set(), index=48)
    bpy.ops.mesh.select_linked_pick(deselect=False, delimit=set(), index=57)
    bpy.context.object.active_material_index = 1
    bpy.ops.object.material_slot_assign()
    bpy.ops.object.mode_set(mode='OBJECT')

class MeuOperador(bpy.types.Operator):
    bl_idname = "meusops.felpudo"
    bl_label = "Meu Operador"

    def execute(self, context):
        main(context)
        return{'FINISHED'}

def register():
    bpy.utils.register_class(MeuOperador)

def unregister():
    bpy.utils.unregister_class(MeuOperador)

register()
```

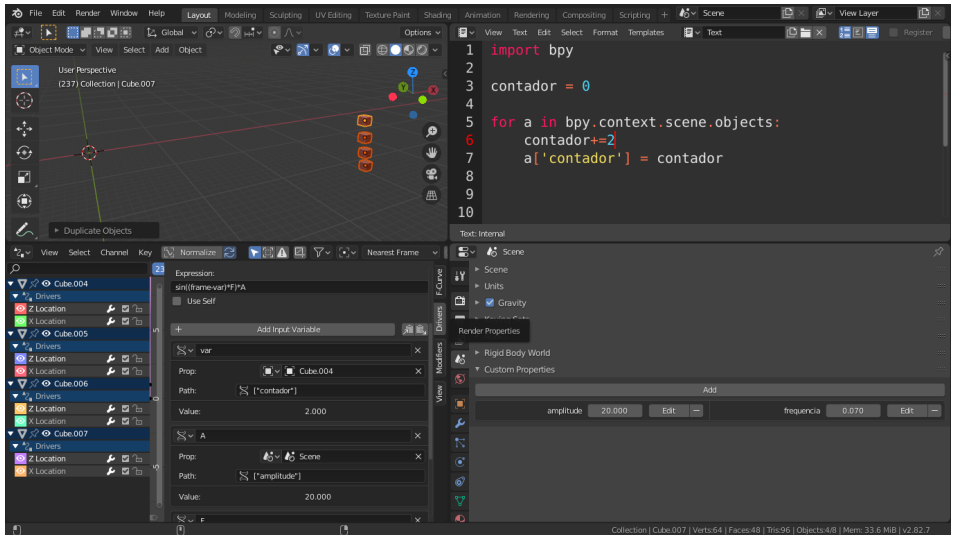
Utilizando o seu Próprio Operador

```
import bpy

# Apaga todos os Objetos da Cena
bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete(use_global=False)

# Apaga todos os Materiais da Cena
materiais = bpy.data.materials
for m in materiais:
    bpy.data.materials.remove(m)

# Loop para Testar Novo Operador
for i in range(10):
    bpy.context.scene.cursor.location.x = i*2
    bpy.ops.meusops.felpudo()
```

Como executar Automaticamente um Script ao abrir o Blender

Basta deixar o script renomeado com a extensão .py dentro da pasta startup dentro de uma pasta apontada nas preferências em File Paths > Scripts.

Como criar um Add-On

```
bl_info = {
    "name" : "Add-on do Felpudo",
    "author" : "Tito Petri",
    "version" : (1,0),
    "blender" : (2,81,16),
    "location" : "View3D > Object",
    "description" : "Adiciona operador Smooth ao objeto. Atalho: Shift+U"
}

import bpy

from bpy.types import (
    AddonPreferences,
    Operator,
    Panel,
    PropertyGroup
)
```

```
class FelpudoSmooth(Operator):
    bl_label = "Felpudo Smooth Tool"
    bl_idname = "object.felpudosmooth"
    bl_description = "Suavizador de Poligonos do Felpudo"
    bl_space_type = "VIEW_3D"
    bl_region_type = "UI"
    bl_options = {"REGISTER","UNDO"}

    def execute(self,context):
        # Aqui vai o seu Script inteiro
        bpy.ops.object.shade_smooth()
        return {'FINISHED'}

def meuMenuFelpudo(self, context):
    self.layout.operator(FelpudoSmooth.bl_idname)

def register():
    bpy.utils.register_class(FelpudoSmooth)
    bpy.types.VIEW3D_MT_object.append(meuMenuFelpudo)

    atalhos = bpy.context.window_manager.keyconfigs.addon
    if atalhos:
        tecla = atalhos.keymaps.new(name="3D View", space_type="VIEW_3D")
        atalho_item = tecla.keymap_items.new('object.felpudosmooth', 'U', 'PRESS', shift=True)

def unregister():
    bpy.utils.unregister_class(FelpudoSmooth)
    bpy.types.VIEW3D_MT_object.remove(meuMenuFelpudo)

if __name__ == "__main__":
    register()
```

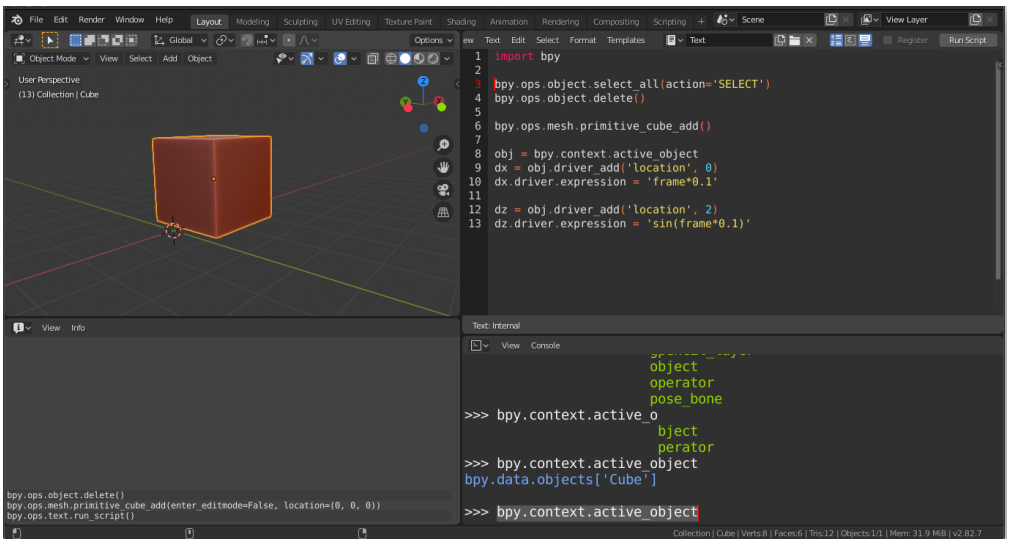
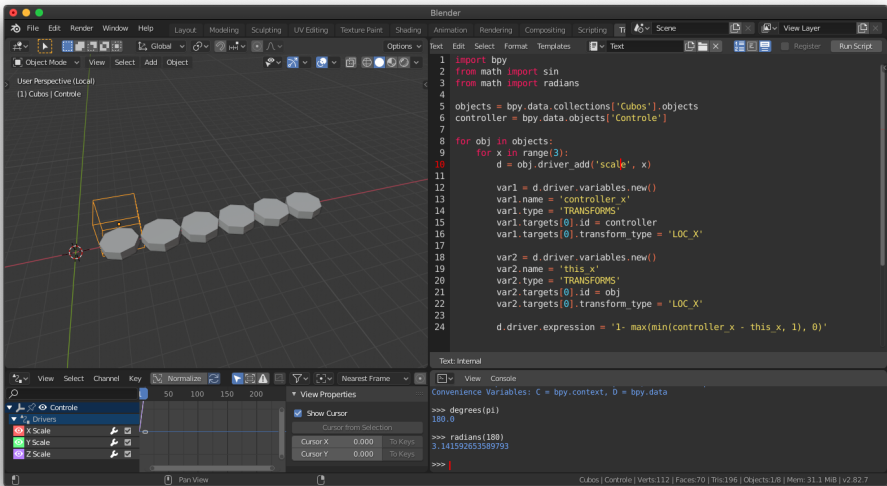
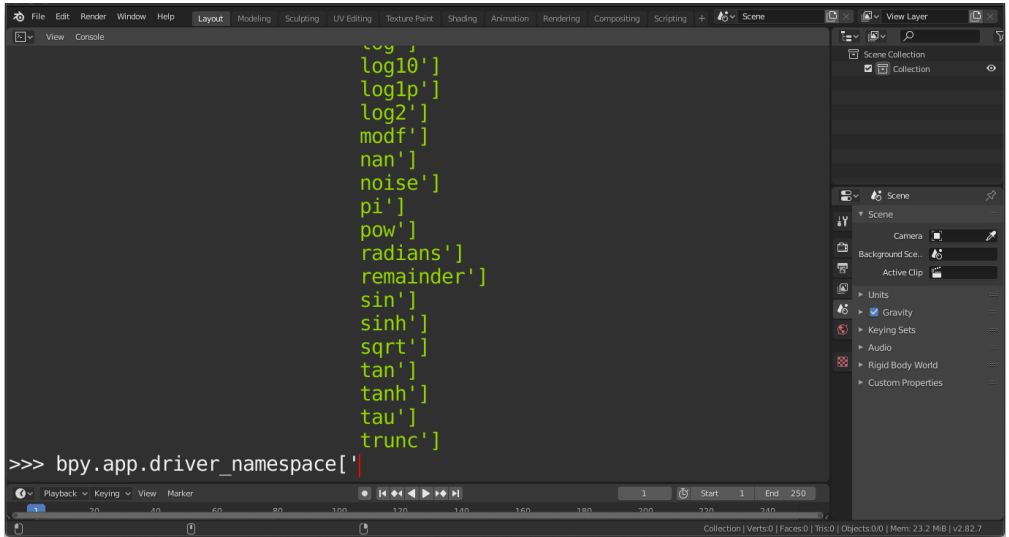
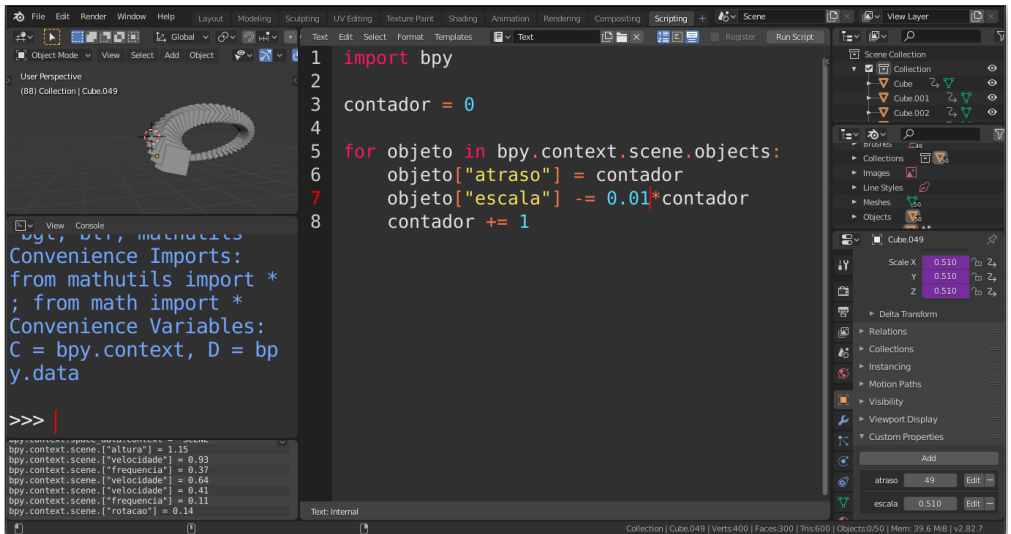
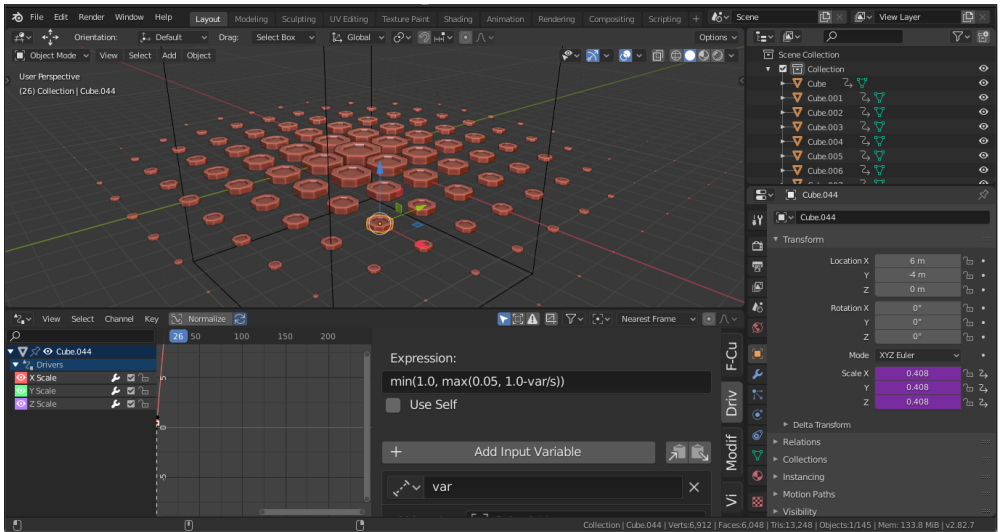
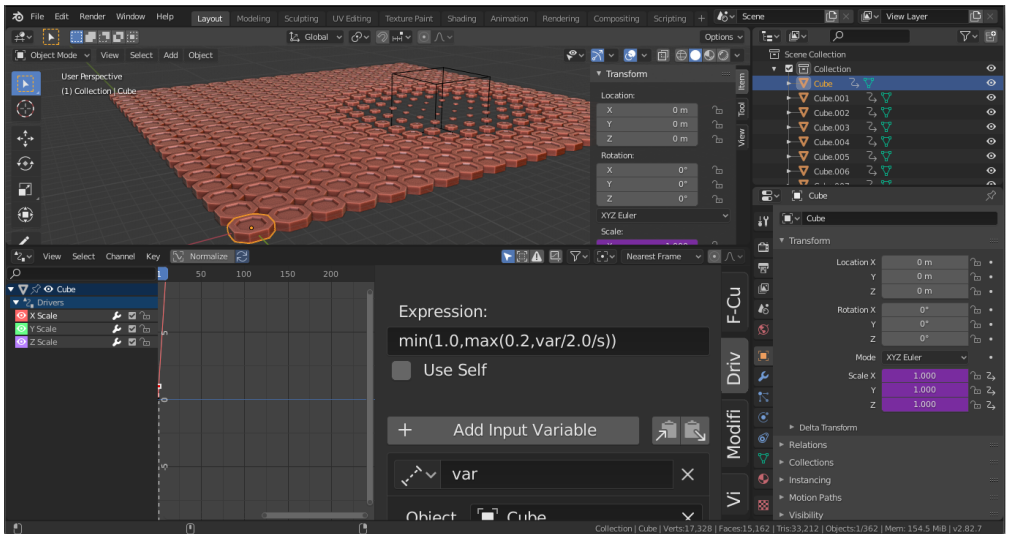
Blender Preferences
Interface
Themes
Viewport
Lights
Editing
Animation
Add-ons
Input
Official
Community
Testing
Install...
Refresh
Enabled Add-ons Only
All
Add-on do Felpudo
: Add-on do Felpudo
Description:
Adiciona operador Smooth ao objeto. Atalho: Shift+U
Location:
View3D > Object
File:
/Users/mac/Library/Application Support/Blender/2.82/scripts/addons/Meu Add..
Author:
Tito Petri
Version:
1.0
Internet:
Remove

Exercícios com Drivers e Expressões

Adicionando Drivers Programaticamente

```
import bpy
from math import sin
from math import radians

objects = bpy.data.collections['Cubos'].objects
controller = bpy.data.objects['Controle']

for obj in objects:
    for x in range(3):
        d = obj.driver_add('scale', x)

        var1 = d.driver.variables.new()
        var1.name = 'controller_x'
        var1.type = 'TRANSFORMS'
        var1.targets[0].id = controller
        var1.targets[0].transform_type = 'LOC_X'

        var2 = d.driver.variables.new()
        var2.name = 'this_x'
        var2.type = 'TRANSFORMS'
        var2.targets[0].id = obj
        var2.targets[0].transform_type = 'LOC_X'

        d.driver.expression = '1- max(min(controller_x - this_x, 1), 0)'
```

Exemplo de adicionar variavel no driver

```
import bpy

bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()

bpy.ops.mesh.primitive_cube_add()

obj = bpy.context.active_object
dx = obj.driver_add('location', 0)
dx.driver.expression = 'frame*0.1'

dz = obj.driver_add('location', 2)
dz.driver.expression = 'sin(frame*0.1)'
```

```
>>> bpy.context.active_object
bpy.data.objects['Cube']
```

```
import bpy

contador = 0

for objeto in bpy.context.scene.objects:
    objeto["atraso"] = contador
    objeto["escala"] -= 0.01*contador
    contador += 1
```

Parabéns querido aluno por chegar até aqui e ter adquirido mais este valioso conhecimento!

Se quiser aprender sempre mais sobre criação de Jogos e Aplicativos, não deixe de conhecer o Aprenda Programar, meu portal de cursos online onde você pode se especializar em:

- Algoritmos e Lógica de Programação
- Modelagem e Animação 3D
- Criação de Personagens para Jogos e Filmes
- Programação de Aplicativos Nativos para iOS e Android
- Criação de Games 2D, 3D e Realidade Virtual
- Realidade Aumentada e Visão Computacional
- Metodologia STEAM
- Robótica e Impressão 3D

Para virar aluno do Aprenda Programar você deve se inscrever pela plataforma Hotmart, no link abaixo.

Adquira seu acesso para sempre ao Aprenda Programar:
https://hotmart.com/product/en/aprenda-programar-com-tito-petri